Reconstructing persistent graph structures from noisy images

Alexey Chernov[1], Vitaliy Kurlin[2]

[1] Dept Computer Science and Technology, University of Bedfordshire, Luton, LU1 3JU, UK, and Computer Learning Research Centre, RHUL, Egham, Surrey, TW20 0EX, UK.
[2] Department of Mathematical Sciences, Durham University, Durham DH1 3LE, UK.
chernov@cs.rhul.ac.uk, vitaliy.kurlin@durham.ac.uk

Abstract Let a point cloud be a noisy dotted image of a graph on the plane. We present a new fast algorithm for reconstructing the original graph from the given point cloud. Degrees of vertices in the graph are found by methods of persistent topology. Necessary parameters are automatically optimized by machine learning tools.

Keywords: graph reconstruction, noisy image, point cloud data, persistent topology

1 The problem of a graph reconstruction

We build up on the methods for a graph reconstruction in [1]. Our goal is to reduce the number of arbitrarily chosen or manually adjustable parameters and to avoid any *ad hoc* preprocessing when there is no metric graph structure on the data.

Let us consider a black-and-white dotted image or a collection of black pixels on a white background. We assume that a point cloud C of black pixels represents a noisy image of a graph $\Gamma \subset \mathbb{R}^2$. Each edge is a polygonal line of straight arcs.

We prove that our algorithm reconstructs the graph Γ with a high probability under reasonable restrictions on Γ and a point cloud $C \subset \mathbb{R}^2$ sampled around Γ.

2 Methods of persistent topology

Persistent topology is a new branch of computational topology that studies topological features (homology groups) of a filtration (a collection of spaces). Starting from a point cloud $C \subset \mathbb{R}^2$, let us define the filtration of graphs $\{C(\epsilon)\}$ as follows.

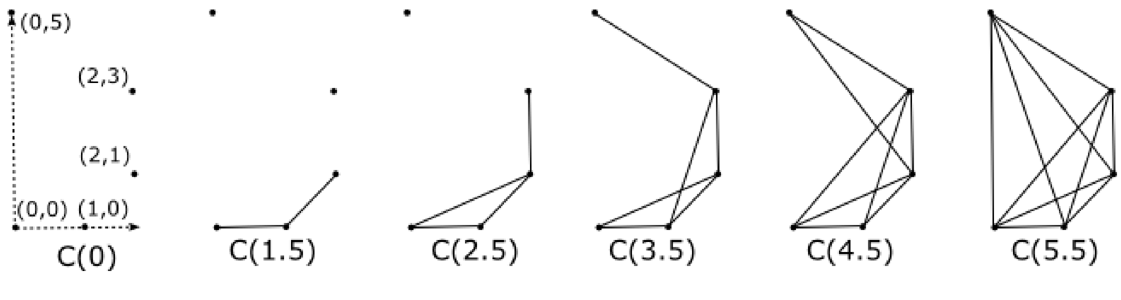

Figure 1: some graphs $C(\epsilon)$ for the point cloud C of 5 points $(0,0)$, $(1,0)$, $(2,1)$, $(2,3)$, $(0,5)$

The vertices of any graph $C(\epsilon)$ are all points from the cloud C. For a threshold distance $\epsilon \geq 0$, if two points $p, q \in C$ are at distance not more than ϵ, we connect p, q by an edge in the graph $C(\epsilon)$. Hence, for any small enough $\epsilon \geq 0$, the graphs $C(\epsilon)$ consist of only isolated vertices at the points of the cloud C, see Fig. 1.

However, when ϵ is increasing, we add more edges and eventually $C(\epsilon)$ becomes a complete graph where any two vertices are connected by an edge. The resulting filtration $\{C(\epsilon)\}$ characterizes relative positions of points in the original cloud C.

Now we can compute topological invariants of $C(\epsilon)$ (for example, the number of connected components or the number of independent cycles) and analyze how they change as the parameter ϵ increases. For instance, if the graphs $C(\epsilon)$ have two connected components over a long range of ϵ, then the original cloud C is likely to have two clusters. Similarly, counting cycles in $C(\epsilon)$ may reveal that the point cloud C looks like a topological closed loop. See [2] for a much more general discussion.

3 First stage: split points by degree

In the first stage of the algorithm, we split a given point cloud C into several parts:

- collections of *edge points*, which will form edges of a future reconstructed graph G;
- collections of *vertex points*, which will form vertices (of different degrees) in G.

To put each point $p \in C$ into one of the collections above, we find a number $\deg_C(p)$ that resembles the degree of a vertex in a graph. Recall that the *degree* of a vertex v in a graph Γ is the number of arcs attached to v, see Fig. 2. Points on edges between vertices of Γ can be considered as trivial vertices of degree 2.

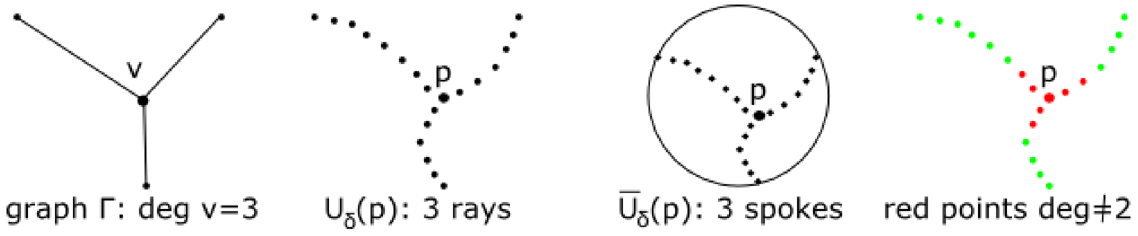

Figure 2: finding a degree of a point p, points of potential degree 2 are green, others are red

We can not count arcs attached to a point p in a cloud C and adapt another way of finding degrees. If a sufficiently small neighborhood $U(v) \subset \Gamma$ of a vertex $v \in \Gamma$ is a 'star with k rays', then $\deg v = k$. For each point $p \in C$, we consider all points from the cloud C within the neighborhood $U_\delta(p)$ with center p and radius δ.

The authors of [1] count the number of small clusters of $C \cap U_\delta(p)$ near the boundary of $U_\delta(p)$ to get the potential degree $\deg_C(p)$. We add the boundary circle to $U_\delta(p)$ and count 'holes' (independent cycles in the first homology) in the 'wheel' $C \cap \bar{U}_\delta(p)$, which seems more resilient to the noise than counting small clusters.

So we convert the small cloud $C \cap \bar{U}_\delta(p)$ into the filtration of 2-dimensional Rips complexes $R(\epsilon)$, which are triangulated subsets of the plane. Namely, the vertices of $R(\epsilon)$ are all points of the small cloud. Two vertices are connected by an edge if the distance between them is not more than ϵ. If any two edges intersect, then we drop one of them. Three vertices span a triangle in $R(\epsilon)$ if they are pairwise connected. The number of independent cycles in the small triangulation $R(\epsilon)$ can be found fast and equals the number of 'holes' whose diameter is more than ϵ.

The resulting number $\deg_C(p)$ depends on a radius δ and a threshold distance ϵ. We shall later show that under certain conditions on Γ and C the degree $\deg_C(p)$ remain stable when δ and ϵ vary over long enough intervals for many points $p \in C$. So our algorithm automatically learns the parameters δ and ϵ for any point $p \in C$.

4 Second stage: vertex clusters and edge clusters

After finding potential degrees $\deg_C(p)$ for all points $p \in C$, we split C into several smaller subclouds C_k by the degree k. Our next aim is to divide the subcloud C_2 of all points p with $\deg_C(p) = 2$ into clusters that represent edges of the graph G.

We consider the filtration of graphs $C_2(\epsilon)$, where the vertices are all points $p \in C$ with $\deg_C(p) = 2$. Two vertices of the graph $C_2(\epsilon)$ are connected by an edge if the distance between them is not more than ϵ. We shall later show that under certain conditions on G and C, some connected components of $C_2(\epsilon)$ remain stable over a long interval of ϵ and can represent edges of the underlying graph G.

Similarly, for any $k \neq 2$, we split the subcloud C_k of all points $p \in C$ with $\deg_C(p) = k$ into clusters that represent degree k vertices of G. Under certain conditions on a graph G and a cloud C around G, every vertex of G has a neighborhood where all points from C have $\deg_C(p) = k$ and form a cluster in the subcloud C_k.

In practice, after trying different threshold distances ϵ to form clusters of vertices and clusters of edges, we try to build the corresponding graph where an edge is attached to a vertex if the corresponding clusters have points that are sufficiently close to each other. Then we validate the resulting graph by the following tests:

- every edge joins 2 vertices, so every edge cluster is close to 2 vertex clusters;
- every cluster of points with $\deg_C(p) = k \neq 2$ has exactly k close edge clusters.

To get the final reconstructed graph G, we put a vertex at the center of mass for each vertex cluster and connect all resulting vertices by straight arcs in the plane.

5 Justification of the algorithm: an outline

Input parameters for our algorithm are denoted by Latin letters, e.g. a point cloud C, a number of points n, a fixed point p etc. Greek letters denote parameters that the algorithm learns or does not use, e.g. an original graph Γ, a noise level τ, a threshold distance δ etc. Let $d(p,q)$ be the Euclidean distance between $p, q \in \mathbb{R}^2$.

For a cloud $C \subset \mathbb{R}^2$, denote the least distance between points $p, q \in C$ by $\mathrm{dmin}(C)$. Let $\mathrm{dcon}(C)$ be the least distance ϵ such that the graph $C(\epsilon)$ is connected. The following proposition illustrates the case when an original graph Γ is a set of vertices without edges. Namely, if a given point cloud C splits into subclouds around finitely many unknown centers that are sufficiently away from each other, then the single edge clustering algorithm always correctly identifies the subclouds.

Proposition (single edge clustering). For a set $\nu = \{\nu_1, \dots, \nu_\kappa\} \subset \mathbb{R}^2$, let us take any point cloud $C \subset \cup_{i=1}^{\kappa} U_\tau(\nu_i)$, where $U_\tau(\nu_i)$ is the round disk of a radius τ at the center ν_i. If $\tau < \dfrac{1}{4}\min\Big\{\mathrm{dmin}(\nu) + \mathrm{dmin}(C),\ 2\mathrm{dmin}(\nu) - \mathrm{dcon}(C)\Big\}$, then our clustering algorithm correctly splits the given cloud C into κ expected clusters.

We analyse connected components of the graph $C(\epsilon)$ when ϵ is increasing. In the interval $2\tau < \epsilon < \mathrm{dmin}(\nu) - 2\tau$, all graphs $C(\epsilon)$ have expected κ components. The upper bound on τ guarantees that this interval is long enough. A forthcoming extended paper with proofs will be available by April 2013 at the webpage below.

6 Java applet for graph reconstruction

www.maths.dur.ac.uk/~dma0vk/java-applets/graph-reconstruction.html

The version of the Java applet on 1st March 2013 shows how the algorithm works for random clouds that are uniformly generated around the following graphs:

- a 'star' graph with one central vertex and a random number of 3-8 rays (spokes),
- a 'wheel' graph obtained from a star graph by adding the polygonal boundary,
- a branched cross whose 4 main rays of random length are along regular bisectors in 4 quadrants, each of these rays has 2 smaller tentacles with random endpoints.

The original graph that the algorithm will reconstruct is shown in green. Clicking on the '*new random cloud*' button generates a blue point cloud around a chosen graph. The button '*split points by degree*' calls for the first stage of the algorithm computing the potential degree $\deg_C(p)$ of each point p in the generated point cloud C. Degree 1 points are highlighted in blue, degree 2 points are green, degree 3 points are red, degree 4 points are cyan, points of higher degrees are yellow.

The button '*reconstructed graph*' shows the final red graph with the original green graph. The 'demo' button demonstrates successive stages with a time delay: a new point cloud, all points colored by degree and a reconstructed graph. The applet starts with a demo mode for a wheel graph with 5 vertices and 8 edges.

Acknowledgements

The research was supported by the EPSRC grant EP/I030328/1 *Persistent topological structures in noisy images.* We thank Gunnar Carlsson (Stanford) and Frédéric Chazal (INRIA Saclay) for fruitful discussions at an ealier stage of our research.

References

[1] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. J. Guibas, D. Morozov: Metric Graph Reconstruction from Noisy Data, Int J Comp Geometry Appl, v.22 (2012), n. 4, p. 305–325.

[2] G. Carlsson: Topology and Data, Bull Amer Math Society, v.46 (2009), n. 2, p. 255–308.